(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

# APPLICATION OF A BACK-PROPAGATION NEURAL NETWORK TO ISOLATED-WORD SPEECH RECOGNITION

by

Chau Giang Le

June 1993

Thesis Advisor: Murali Tummala

| REPORT DOCUMENTATION PAGE | Form Approved OMB No. 0704-0188 |
|---|---|

| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | Jun 93 | Master's Thesis |

4. TITLE AND SUBTITLE

APPLICATION OF A BACK-PROPAGATION NEURAL NETWORK TO ISOLATED-WORD SPEECH RECOGNITION

5. FUNDING NUMBERS

6. AUTHOR(S)

LE, Chau Giang

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the U.S. Government or the Department of Defense.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public resease; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The primary objective of this research is to explore how a back-propagation neural network (BNN) can be applied to isolated-word speech recognition. Simulation results show that a BNN provides an effective approach for small vocabulary systems. The recognition rate reaches 100% for a 5-word system and 94% for a 10-word system. The general techniques developed in this thesis can be further extended to other applications, such as sonar target recognition, missile seeking and tracking functions in modern weapon systems, and classification of underwater acoustic signals.

14. SUBJECT TERMS

BNN (back-propagation neural network) isolated-word speech

15. NUMBER OF PAGES

45

16. PRICE CODE

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLAS | UNCLAS | UNCLAS | UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std 239-18

# Application of a Back-propagation neural network to isolated-word speech recognition

by

Chau Giang Le
Lieutenant, United States Navy
B.S., University of California, San Diego, 1985

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1993

Author: [illegible]
Chau Giang Le

Approved by: [illegible]
Murali Tummala, Thesis Advisor

[illegible]
Charles W. Therrien, Second Reader

[illegible]
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

## ABSTRACT

The primary objective of this research is to explore how a back-propagation neural network (BNN) can be applied to isolated-word speech recognition. Simulation results show that a BNN provides an effective approach for small vocabulary systems. The recognition rate reaches 100% for a 5-word system and 94% for a 10-word system. The general techniques developed in this thesis can be further extended to other applications, such as sonar target recognition, missile seeking and tracking functions in modern weapons systems, and classification of underwater acoustic signals.

# TABLE OF CONTENTS

# I. INTRODUCTION

The demand for real time processing in a vast majority of applications (e.g., signal processing and weapon systems) has led researchers to seek novel approaches to solve the conventional serial processing bottleneck. Artificial neural networks, based on human-like perception characteristics, provide one such approach that is currently receiving much attention. Due to the significant progress in the computer field and new technology, neural networks, once considered a dead area, are resurrecting from a four-decade "dormant" period.

The primary objective of this thesis research is to explore how neural networks can be employed to recognize isolated-word speech as an alternative to the traditional methodologies. The main benefit of this study would be its contribution toward understanding the neural network-based techniques for solving the common but difficult problem of pattern recognition, particularly in automatic speech recognition (ASR). This technique can be further extended to various other important practical applications, such as sonar target recognition, missile seeking and tracking functions in modern weapon systems, and classification of underwater acoustic signals.

The thesis is organized as follows. Chapter II introduces neural networks. Reasons for using neural networks in speech recognition are briefly discussed, and the structure and characteristics of a back-propagation neural network (BNN) are described. Chapter

III reviews basics of speech recognition and describes techniques to estimate the parameters that are used as features of the speech signal in the recognition step. Chapter IV presents the results of a small vocabulary BNN system. Chapter V summarizes the important results, discusses the limitations of the proposed BNN system, and offers ideas for further research.

# II. NEURAL NETWORKS

## A. WHY NEURAL NETWORKS?

Recently, investigations have revealed the potential of neural networks as useful tools for various applications that require extensive categorization. Several applications of neural networks have been proven successful or partially successful in the areas of character recognition, image compression, medical diagnosis, and financial and economic prediction [Ref. 1]. The power of parallel processing in neural networks and their ability to classify the data based on selected features provide promising tools for pattern recognition in general and speech recognition in particular.

Traditional sequential processing technology has serious limitations for implementing pattern recognition related problems. The classical approach for pattern recognition is often costly, inflexible, and requires data intensive processing [Ref. 2].

On the other hand, neural networks accomplish the processing task by training instead of programming in a manner analogous to the way the human brain learns. Unlike traditional Von Neumann sequential machines where formula and rules need to be specified explicitly, a neural network can derive its functionality by learning from the examples presented.

## B. ARCHITECTURE

Neural networks consist of a large number of neurons or processing elements, PE for short. Each PE has a number of inputs with associated connection weights as shown in Figure 1. These weighted inputs are summed and then mapped through a nonlinear threshold function. The threshold function, also called an activation function or transfer function, is continuous, differentiable and monotonically increasing [Ref. 1]. Two widely used threshold functions are the sigmoid

$$f(z) = \frac{1}{1+e^{-z}}, \tag{2-1}$$

and the hyperbolic tangent

$$f(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}. \tag{2-2}$$

The processing elements are arranged in layers in a predefined manner. Each PE processes the inputs it receives via its weighted input connections and provides a continuous value to other PEs via its outgoing connection.

The basic neuron model shown in Figure 1 can be easily realized in hardware. An equivalent electrical analog model could be built using an operational amplifier with non-linear feedback to represent the summing operator and the activation function while the connection weights could be realized by the input resistors to the operational amplifiers [Ref. 3].

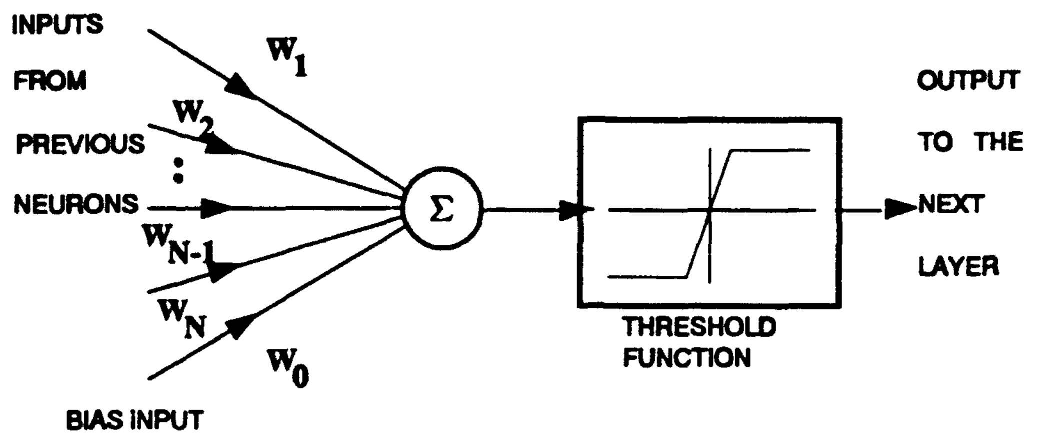

**Figure 1.** An artificial neuron.

## C. BACK PROPAGATION NEURAL NETWORKS

Currently, the most popular and commonly used neural network is the back-propagation neural network (BNN). A typical back-propagation neural network is a feed forward multi-layer network with an input layer, an output layer, and at least one hidden layer. Each layer is fully connected to the succeeding layer; however, there is no inter-connection or feedback between PEs in the same layer. Each PE has a bias input with an associated non-zero weight. The bias input is analogous to the ground connection in an electrical circuit in that it provides a constant input value. The data flow from the input layer to the output layer. Figure 2 shows a typical multi-layer back-propagation network with a single hidden layer.

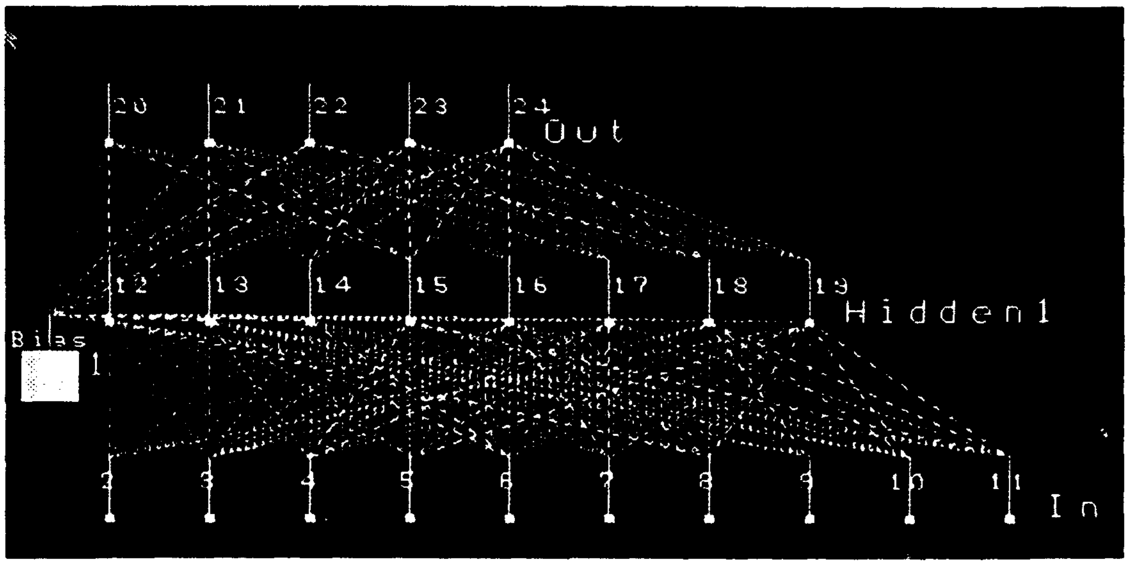

Figure 2. A back-propagation neural network

In a neural network consisting of N processing elements, the input/output function is defined as [Ref.4]:

$$y = G(x, W) \tag{2-3}$$

where $x = \{x_j\}$ is the input vector to the network, $y = \{y_i\}$ is the output vector from the network, and **W** is the weight matrix. The latter is defined as

$$W = (w_1^T, w_2^T, \ldots, w_N^T)^T \tag{2-4}$$

where the vectors $w_1, w_2, \ldots, w_N$ are the individual PE weight vectors, which are given by

$$w_i = \begin{bmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{in} \end{bmatrix}, \qquad i = 1, 2, .., N. \tag{2-5}$$

These connection weights adapt or change in response to the example inputs and the desired outputs according to a learning law. Most learning laws are formulated to guide the weight vector $w$ to a location that yields the desired network performance. There exist many learning laws depending on the type of application: delta rule or LMS algorithm, competitive learning, and adaptive resonance theory to name a few [Ref. 4].

The back-propagation neural network implements the delta rule to update the connection weights. The delta rule is a modification of the least mean square (LMS) algorithm in which the global error function E between the current output, $y_i$, and the desired output, $d_i$, given by

$$E = \frac{1}{2}\sum_j (d_j - y_j)^2, \tag{2-6}$$

is minimized. The resulting weight update equation can be shown to be [Ref. 1]

$$w^s_{ij\,new} = w^s_{ij\,old} + \mu\left(\frac{\partial E}{\partial w_{ij}}\right) + \nu\ (w^{s-1}_{ij\,new} - w^{s-1}_{ij\,old}) \tag{2-7}$$

where $\mu$ is the learning rate, $\nu$ is the momentum term, and $s$ is the number of the layer.

The learning rate $\mu$ is equivalent to the step-size parameter in the LMS algorithm. Selection of an appropriate value for $\mu$ is usually done by trial and error. Too large a learning rate often leads to divergence of the learning algorithm while too small a value results in a slow learning process. The value of $\mu$ is required to be in the range $0 < \mu < 1$ [ Ref. 4]. The momentum term $v$ was added in equation (2-7) to act as a low pass filter on the delta weight term; it is used to help "smooth" out the weight changes. The value of $v$ usually lies between zero and one [Ref. 1]. The use of a momentum term has been found to speed up the learning process considerably. It allows a larger value for $\mu$ yet avoids divergence of the algorithm.

The main mechanism of the back-propagation neural network is to propagate the input through all layers to the output layer. At the output layer, the errors are determined and the associated weights, $w_{ij}$, are updated by equation (2-7). These errors are then propagated back through the network from the output layer to the previous layer (thus the name back-propagation). The back-propagated error at PE j in layer s is given by [Ref. 1.]

$$e_j^{[s]} = x_j^{[s]} \, (1.0 - x_j^{[s]}) \sum_k (e_k^{[s+1]} \, w_{kj}^{[s+1]}). \qquad (2\text{-}8)$$

This process is continued until the input layer is reached.

In summary, Artificial Neural Networks, which use highly parallel architectures with the processing distributed among the PEs, overcome the limitation of the conventional sequential processing bottleneck. The back-propagation algorithm discussed in this chapter provides a simple mechanism to implement a nonlinear mapping between input and output.

# III. BASICS OF SPEECH RECOGNITION

## A. EXISTING DIFFICULTIES IN ISOLATED-WORD SPEECH RECOGNITION

Although the field of automatic speech recognition has been in existence for several decades, many difficulties of ASR still exist and need to be addressed [Ref. 5]. Besides the syntactic and semantic issues in the linguistic theories, speech segmentation is a big concern. Boundaries between words and phonemes are very difficult to locate except when the speaker pauses. There is no separation between words as compared with written language. Although boundaries can be estimated by a sudden large variation in the speech spectrum, this method is not very reliable due to coarticulation, i.e., the changes in the articulation and acoustics of a phoneme due to its phonetic content. The speech signal is context dependent, i.e., the word before and the word after affect how the signal is formed. The speech signal is also dependent on the speaking mode. Softness, loudness, and stuttering also affect the speech signal. The environment, such as a quiet room, background noise, or co-channel interferences, effects the signal tremendously. The input devices, such as microphone, amplifier, and recorder, also alter the speech signal [Ref.6].

Due to the asymmetries in producing and interpreting it, speech is a complex signal for recognition. For effective results, ASR requires an approach that is closer to human perception, and traditional computing techniques are inefficient at such tasks. Neural networks, which are modeled after the human brain, seem to provide a useful tool for speech recognition.

## B. BASICS OF SPEECH RECOGNITION

Speech data are often redundant and too lengthy to apply the whole signal to the recognizer. A pre-processing step is often required to achieve high efficiency. This step involves determining the discriminating characteristics that are particular to the signal [Ref. 2]. Most speech recognizers utilize only the aspects that help in sound discrimination and capture spectral information in a few parameters to identiiy the spoken words or phonemes. Typical speech signal parameters that are used as features are LPC coefficients, energy, zero-crossing rate and voiced/unvoiced classification.

Most ASR systems utilize the same techniques as those used in the traditional pattern recognition area. The procedure to implement these systems involves several steps: normalization, parameterization, feature extraction, similarity comparison, and decision. Figure 3 shows a block diagram of the general procedure for speech recognition [Ref.7]. The speech signal is fed into a pre-processor which performs the amplitude normalization, parameterization and generates a test template based on these features. The test template is then compared with a set of pre-stored reference templates. The reference template that most closely matches the test template is determined based on predetermined decision rules.

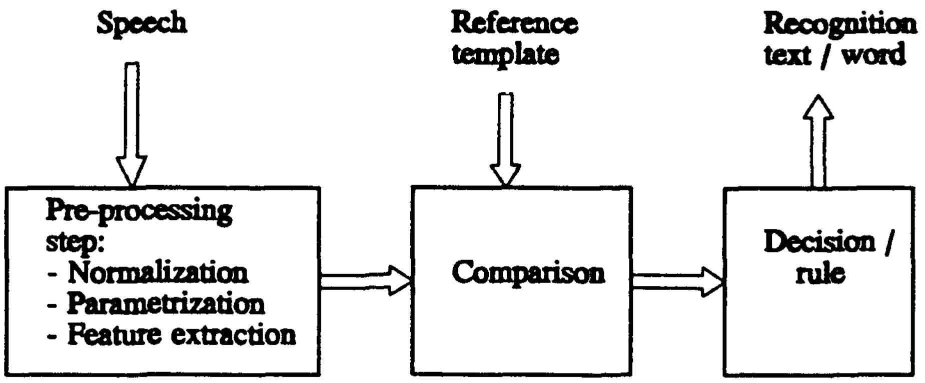

**Figure 3.** Pattern Recognition block diagram for ASR.

### 1. Normalization

The magnitude normalization step attempts to eliminate the variability in the input speech signal due to environment --- variations in recording level, distance from the microphone, original speech intensity, and loss of signal in transmission.

### 2. Parameter estimation

The parameterization step involves converting the speech signal into a set of statistical parameters that maximize the likelihood of choosing the right output features corresponding to the input signal. Since most of the useful speech acoustic details are below 4 KHz, a bandwidth of 3 to 4 KHz is sufficient to represent the speech signal.

Experience has shown that using extra bandwidth actually deteriorates the performance of the ASR [Ref. 5]. In general, eight to fourteen LPC coefficients are considered sufficient to model the spectral behavior of the short-time speech spectral envelope [Ref.3]. Other parameterization techniques, such as cepstral analysis, can also be used. The test template is formed by an arrangement of the features, such as LPC or Ceptral coefficients, short-time energy, short-time zero-crossing rate, and V/U switch.

### 3. Similarity comparison and decision

Even for a single speaker, the same utterances are spoken with different durations and at different speaking rates when repeated; therefore, a normalization of the reference utterance along the time axis is needed before the comparison step in figure 3 [Ref.5]. Linear time warping, which is accomplished through frame interval adjustment before parameterization or through decimation/interpolation of the feature sequence, can be used to partially overcome this problem. Linear time warping is simple to implement; however, it ignores the non-linearity of the speaking-rate variation. For example, the energy of long and short versions of the same word does not distribute evenly for each phoneme but is often concentrated on a dominant vowel. Other approaches append silence to the ends of the shorter utterances, using the length of the longest utterance as the reference. The test template is then compared with the reference templates based on the Euclidian distance or LPC distance measure. The word corresponding to the reference template that yields the minimum distance is announced as the word being spoken.

When the ASR is accomplished by neural networks, the comparison and decision steps are performed through training. The neural network is trained by presenting examples of the reference templates. A learning algorithm based on the delta rule discussed in chapter two is used to minimize a global error function between the current output and the desired output. Once the network is "trained", i.e., the error or the delta weight term ($w_{ij_{new}}$ - $w_{ij_{old}}$) is less than a pre-set threshold, a final set of weight values are obtained and used as the network characteristics. The unknown (or test) signal is then passed through this network with fixed weight values; the word corresponding to the highest value in the output vector is the word being recognized. The recognition rate is then calculated based on the percentage of correctly identified outputs.

In short, understanding the techniques and related difficulties in conventional speech recognition will help identify the key parameters to form the feature vector for the back-propagation based speech recognition system reported here.

# IV. SOFTWARE SIMULATIONS STUDIES AND RESULTS

## A. DEVELOPMENT OF SIMULATION

In this section, the back-propagation neural network is applied to speaker independent speech data extracted from the TIMIT database. Several experiments were conducted using NeuralWorks Professional II/Plus[1] software, covering the effects of different parameters and feature vectors that are discussed later in the chapter.

### 1. Timit database

The speech data used in this thesis, for both training and testing, were extracted from the TIMIT database. The data was prepared by the National Institute of Standards and Technology (NIST) with sponsorship from the Defense Advanced Research Project Agency's Information Science and Technology Office (DARPA-ISTO). Text corpus material design was a joint effort among the Massachusetts Institute of Technology, Stanford Research Institute, and Texas Instruments [Ref.8].

TIMIT contains a total of 6300 sentences spoken by 630 speakers from 8 different major geographical areas. The text material consists of phonetically-compact sentences as well as phonetically diverse sentences to provide a good coverage of phoneme utterances and add diversity to the phonetic contexts. Each sentence is broken down into four different representations: waveform, text, word, and phoneme. The

[1] NeuralWorks Professional II/Plus® is a registered trademark of NeuralWare, Inc.

waveform is the speech data in the original digitized format. The text is the orthographic transcription of the words. Word and phoneme representations use time-aligned word and phoneme transcription, respectively. The boundaries were aligned with the phonetic segments using dynamic string alignment program. For more details, refer to the documentation of the TIMIT database [Ref. 8].

The original continuous speech database files were first converted into the European multi-lingual speech input/output assessment methodology and standardization (SAM) speech database format and then into the ASCII format. Once in this from, they can be processed using MATLAB[2].

### 2. NeuralWorks Professional II/Plus software

The results of the experiments conducted in this thesis are derived from NeuralWorks, a simulation software program from NeuralWare, Inc.

> NeuralWorks is a comprehensive multi-paradigm prototyping and development system used to design, build, train, test and deploy neural networks to solve complex, real world problems. It allows [the user] to easily generate over 20 well known network types, or design ... custom networks. [One] can display the networks through network diagrams or Hinton diagrams through an extensive instrumentation package. A set of powerful diagnostic capabilities let [the user] monitor and display weights, errors and activation levels in graphical formats.

### 3. Speaker independent system

In this research, three speech recognizers, consisting of a 3-word, 5-word and 10-word vocabulary, were developed using a back-propagation neural network. These

[2] MATLAB® is a registered trademark of The MathWorks, Inc.

recognizers were trained using speech recordings of a variety of speakers, resulting in a speaker-independent system. Figure 4 shows the block diagram of a typical 10-word recognizer. The speech signal is passed through an end point detection block to isolate the words. The input vector is then formed by computing the necessary features and fed into the back-propagation neural network. The output vector is first normalized and then compared with the desired output vector, e.g., (1,0,0,0,0,0,0,0,0,0) for the first word, (0,1,0,0,0,0,0,0,0,0) for the second word, and so on. The recognition rate is calculated based on the percentage of correctly matched outputs.

All pattern recognition tasks, including ASR, utilize two phases: training and testing or recognition. Although there are some general guidelines for the size of the training and testing data, in particular problems one is often limited by the available data. Larger data sets produce more reliable results with lower variances on the estimates of performance (error rate). The training set used for the 10-word recognizer includes 100 words spoken by 10 male speakers from one major geographic region. There are two testing sets: one includes 100 words spoken by 10 speakers and the second includes 380 words spoken by 38 different speakers. For the 5-word and 3-word recognizer, the number of speakers remains the same, but the total number of words for the training data reduces to 50 and 30, respectively. The testing data consist of 190 words (5x38) for the 5-word recognizer and 114 words (3x38) for the 3-word recognizer.

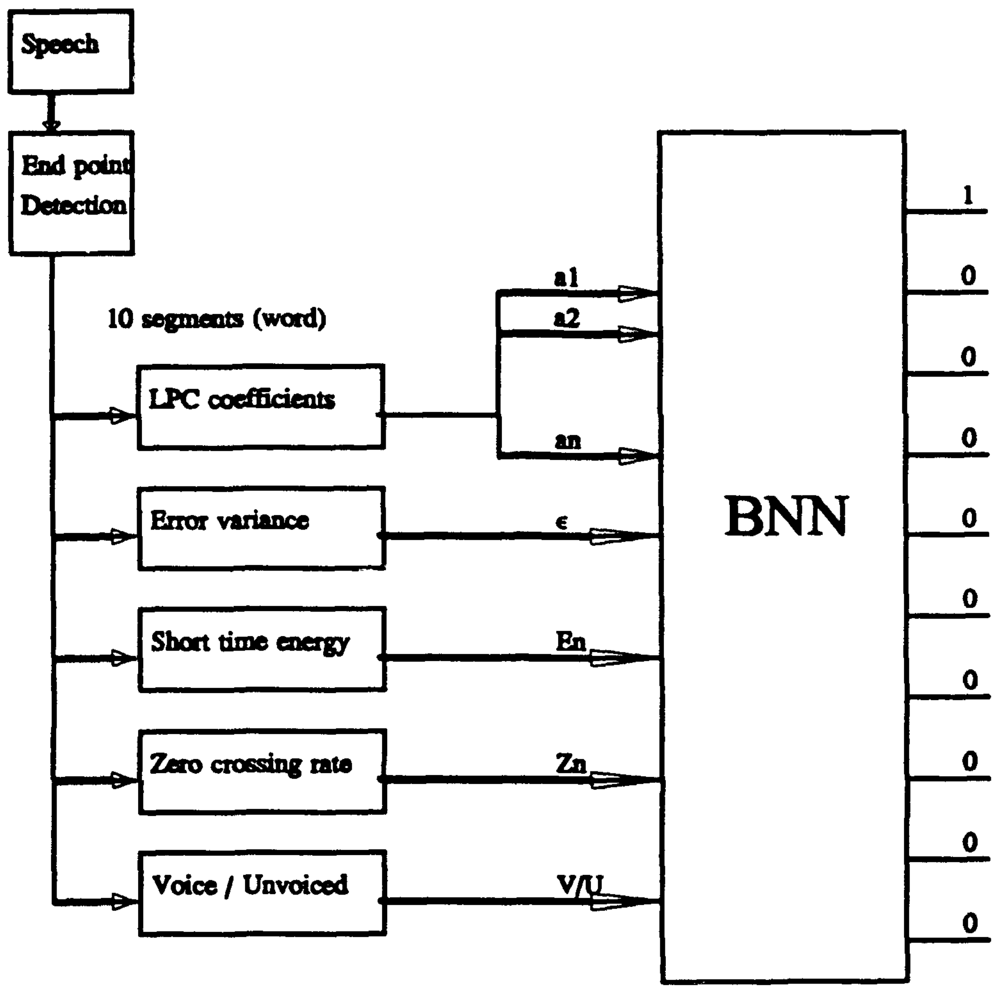

Figure 4. A typical 10-word recognizer

## 4. Preprocessing and feature extraction

The original speech signals from the database are digitized at 16 KHz. Since most of the energy for voiced sounds is concentrated below 4 KHz [Ref. 3], the speech signals are low pass filtered to a bandwidth of 4 KHz and decimated to remove the redundancy. The input vector to the back-propagation neural network consists of linear predictive coding coefficients, $a_n$, error variance, $\epsilon$, short time energy function, $E_n$, short time average zero crossing rate, $Z_n$, and voiced/unvoiced discrimination, V/U.

Speech is a non-stationary and time-variant signal. To cope with the temporal variability, short-time processing techniques are used. Short segments of the speech signal are isolated and processed as if they were segments from a signal that is statistically stationary.

There are two methods of segmenting the speech signal. The first one segments the speech signal into fixed length frames. The second divides the speech signal into a fixed number of frames with variable length. The latter was used in this work to deal with the non-uniform word length in the speech recordings. The digitized speech signal is segmented into 10 frames of 10 to 40 milliseconds each depending on the length of a given word in the vocabulary. Each frame is weighted by a hamming window. A brief summary of the processing steps is given in Figure 5 [Ref.9].

The digitized speech signal sampled at 16 KHz is first pre-processed. It is passed through a lowpass filter and decimated to 8 KHz; the signal is then segmented into 10 variable length frames. The feature vector is obtained by computing the LPC coefficients, the error variance, the short time energy, the zero crossing rate and the voiced/unvoiced classification for each segment. The Matlab programs for processing these parameters in the feature vector are listed in Appendix A.

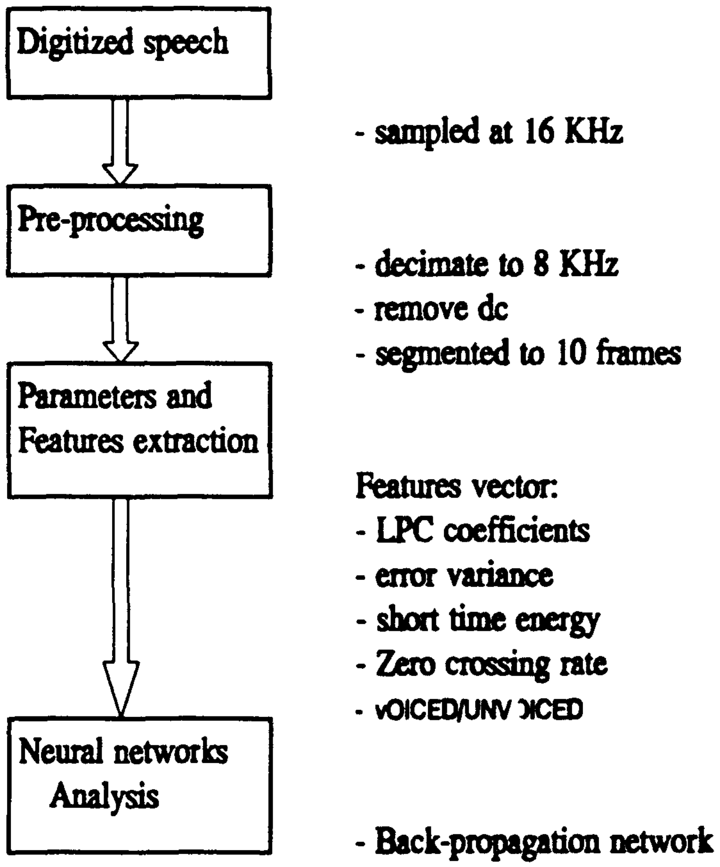

**Figure 5.** Pre-processing block diagram.

***a. LPC coefficients and error variance***

Linear predictive analysis is a technique for estimating the basic speech parameters, e.g., formants, spectrum, and vocal tract area function [Ref. 3]. A unique set of parameters is determined by minimizing the sum of the squared error between the actual and the predicted speech signal. The LPC parameters provide a good representation of the gross features of the short time spectrum. The poles of the transfer function indicate the major energy concentration in the spectrum, and the error variance indicates the energy level of the excitation signal [Ref. 5]. The algorithm used to compute the LPC coefficients is the covariance method which requires solving a set of leastsquare based normal equations; Matlab Code for this is included in Appendix A.

*b. Short-time energy function*

The short-time energy function is used to reflect the amplitude variation between voiced and unvoiced speech segments. This quantity is defined as

$$E_n = \sum_{m=-\infty}^{\infty} x^2(m)h(n-m) \tag{4-1}$$

where $h(n) = w^2(n)$, and $w(n)$ is the Hamming window (a length of 50 is used in this experiment).

*c. Short-time average zero-crossing rate*

The short-time average zero-crossing rate is a simple measure of frequency content of the speech signal. This quantity is defined as

$$Z_n = \sum_{m=-\infty}^{\infty} | \, sgn[x(m)] - sgn[x(m-1)] \, | \, w(n-m) \tag{4-2}$$

where

$$sgn[x(n)] = \begin{cases} 1 & x(n) \geq 0 \\ -1 & x(n) < 0 \end{cases} \tag{4-3}$$

and

$$w(n) = \begin{cases} \frac{1}{2N} & 0 \leq n \leq N-1 \\ 0 & otherwise \, . \end{cases} \tag{4-4}$$

#### *d. Voiced/unvoiced discrimination*

There is a strong correlation between the energy distribution and the voiced/unvoiced classification of speech. The speech signal corresponding to each word is divided into 10 smaller segments for classification as voiced or unvoiced speech. The differentiation between voiced and unvoiced segments is accomplished by setting a threshold based on the short time energy function. For energy values higher than 0.8 of the maximum level, the segment is treated as voiced and assigned a value of 1. For values of energy lower than 0.2 of the maximum level, the segment is classified as unvoiced and assigned a value of 0. Any mid-level is considered intermediate and assigned a value of 0.5.

## B. RESULTS AND ANALYSIS

For consistency and ease of comparison for different cases, BNN parameters are kept constant in all experiments reported here. The learning algorithm used is the delta rule. The transfer function is hyperbolic tangent. The values of momentum and learning rate are 0.2 and 0.5, respectively. The training cycles used are 5000 for 3- and 5-word networks, and 10000 for the 10-word network.

### 1. The feature vector

The performance of a network varies drastically depending on the features used in the training and testing of the network. Too many features may decrease the efficiency of the network since it takes too long to train the network, while too few features may degrade the network's performance.

Two experiments are conducted to emphasize the importance of the choice of the features used in the preprocessing step. The first experiment only uses the LPC coefficients and the error variance. The second one uses all features: LPC coefficients, error variance, short time energy, short time average zero crossing rate, and voiced/unvoiced discrimination.

In the first experiment, a 4th order LPC analysis is performed; for each frame, a set of four LPC coefficients is generated along with the error variance. Thus the input vector to the recognizer consists of a total of fifty parameters per word -- five parameters per frame and ten frames per words.

For the second experiment, the short-time energy, the short-time average zero-crossing rate, and the voiced/unvoiced switch are also computed for each frame, making a total of eighty parameters per word. The results are summarized in Table 1. For a small vocabulary set, the recognition rates do not change much between the two experiments. With the 38-speaker testing set, the recognition rate remains 100% for the 3-word recognizer in both experiments. For the 5-word recognizer the rate decreases from 94.7% to 91.6%. For the 10-word recognizer, the result not only deteriorates from 91.3% to 63.7%, but also the training time increases drastically from 10000 to 80000 training cycles.

TABLE 1. EFFECT OF CHOICE OF FEATURES ON RECONGITION RATE

| Training vector | Vocabulary size | Error | Recognition rate |
|---|---|---|---|
| 50x30 | 3 words | 0 | 100% |
| 50x50 | 5 words | 16/190 | 91.6% |
| 50x100 | 10 words | 138/380 | 63.7% |
| 80x30 | 3 words | 0 | 100% |
| 80x50 | 5 words | 10/190 | 94.7% |
| 80x100 | 10 words | 36/380 | 91.3% |

## 2. LPC order

The order of the LPC coefficients plays an important role in the performance of the network. Four experiments with different LPC orders were conducted on a 10-word vocabulary back propagation network with 10 speaker testing set. A full feature vector is used for all of the following experiments.

TABLE 2. EFFECT OF LPC ORDER ON RECOGNITION RATE

| LPC order | Recognition rate |
|---|---|
| 2 | 84% |
| 4 | 94% |
| 8 | 92% |
| 12 | 86% |

The recognition rates shown in table 2 indicate that the fourth order LPC system captures the essential features of the speech data. The second order system apparently does not have enough parameters to differentiate between the words. However, as the LPC order increases, the complexity of the system also increases, and the information becomes redundant, making the network more cumbersome for training and recognition. This can actually result in decrease recognition rate, as shown.

### 3. Vocabulary size

The size of vocabulary used in the training and testing set also affects the recognition rate of the network. As the number of words used in the network increases, the performance decreases rapidly. The results are summarized in Table 3. With a 3-word vocabulary and a 12th order system, the recognition rate is 96.5% for the 380 word testing set. This rate decreases to 93.2% for a 5-word system and deteriorates to 86.6% for a 10-word system.

TABLE 3: EFFECT OF VOCABULARY SIZE AND LPC ORDER ON PERFORMANCE OF BNN

| Vocabulary size | 3 words | | 5 words | | 10 words | |
|---|---|---|---|---|---|---|
| Testing set size | 100wrds | 380wrds | 100wrds | 380wrds | 100wrds | 380wrds |
| 2nd order LPC | 100% | 100% | 96% | 94.2% | 84% | 87% |
| 4th order LPC | 100% | 100% | 100% | 94.7% | 94% | 91.3% |
| 8th order LPC | 100% | 99.2% | 96% | 94.7% | 92% | 89% |
| 12th order LPC | 100% | 96.5% | 98% | 93.2% | 86% | 86.6% |

### 4. The number of PEs in hidden layers and the number of hidden layers

The structure of the network also affects the performance of the network. A back-propagation neural network generally has one to two hidden layers. The performance of the network increases as the number of PEs increases to an optimal number and then starts to decrease. Here the optimal number of PEs in the hidden layer is determined experimentally; a hidden layer with 24 PEs is found to perform well for the speech data. The single hidden-layer network is found to give better results than the network with two hidden-layers. The results are summarized in tables 4 and 5.

**TABLE 4. EFFECT OF HIDDEN LAYER ON THE RECOGNITION RATE**

| | Testing set size | |
|---|---|---|
| | 100 wrds | 380 wrds |
| one hidden layer 80-24-10 | 94% | 91.3% |
| Two hidden layers 80-24-12-10 | 90% | 87.4% |

**TABLE 5. EFFECT ON THE NUMBER OF PEs IN THE SINGLE HIDDEN-LAYER NETWORK.**

| PE in hidden layer | Testing set size | | Training cycles required |
|---|---|---|---|
| | 100 wrds | 380 wrds | |
| 12 | 91% | 90% | 10000 |
| 24 | 94% | 91.3% | 10000 |
| 32 | 89% | 88.4 | 15000 |
| 64 | 89% | 88 | 25000 |

### 5. Learning rate and Momentum

For a learning rate of 0.5, the effect of the momentum term was experimentally studied and the results are shown in Table 6. A momentum of 0.1 was found to give the best recognition rate and fast training.

TABLE 6. EFFECT OF MOMENTUM ON RECOGNITION RATE

| | Testing set of | | Training cycles |
|---|---|---|---|
| Momentum | 100 wrds | 380 wrds | required |
| .01 | 93% | 90% | 5000 |
| .1 | 97% | 92.1% | 5000 |
| .2 | 94% | 90.5% | 10000 |
| .3 | 90% | 88.4% | 15000 |
| .4 | 73% | 75.8% | 30000 |

### 6. Effect of embedded noise

One special characteristic of the neural network is its non-succeptability to noise. Its ability to classify in a noisy environment, which is demonstrated here, makes it suitable for ASR. Noise added to the speech signal only affects the performance of the network slightly. Uniform random noise may be added to the weighted sum input signal prior applying the weight matrix. The source and the amount of noise added is determined by the mode (learning or testing) and the appropriate parameter in the "Temperature" row from the "learning schedule" [Ref. 1]. Table 7 shows the results of a 10-word vocabulary using a 4th order LPC system with 20% added noise.

TABLE 7. EFFECT OF EMBEDDED NOISE

| | Testing set of 100 wrds | 380 wrds |
|---|---|---|
| No noise | 94% | 90.5% |
| With 20% added noise | 91% | 89.2% |

In summary, the choice of the feature vector in the design of a BNN is very important. The discriminating characteristics that are particular to the speech signal must be recognized to achieve good results. Also, the information fed to the network should be adequate and optimal, i.e., too little information will lead to a low recognition rate, and too much information will result in extensive training time.

# V. CONCLUSIONS

A back-propagation neural network, combined with speech signal processing techniques, is used to develop a speech recognition system. Specifically, a BNN was used to design a 10-word speech recognizer. Experiments were conducted on the recognizer using recorded speech data from the TIMIT database and the simulation software from NeuralWorks. The main observations from these experiments are summarized below:

(1) BNN is an effective approach for small vocabulary ASR. The recognition rate is 100% in most cases for the 3 and 5-word vocabulary systems, and 94% for the 10-word system.

(2) The choice of feature vector plays an important role in the performance of the BNN. The recognition rate may decrease drastically or the system may not converge at all if the features are not correctly chosen. The feature vector chosen in the experiments, which consisted of the LPC coefficients, short time energy, zero-crossing rate and voiced/unvoiced classification, worked well and provided good results for the systems studied.

(3) The techniques developed in this research on isolated-word speech recognition can be extended to other important practical applications, such as sonar target recognition, missile seeking and tracking functions in modern weapon systems, and classification of underwater acoustic signals. However we cannot make predictions about the likely performance of the methods in these areas until they are actually tested.

In this research on ASR, all experiments used male speakers from one specific geographical region. A larger, more diverse group of speakers should be used for a more general case. In addition, the 10-word recognizer system is small for most real applications. Future research should be directed toward larger vocabulary systems, involving say 50 words or more.

The emphasis of the research was to develop an isolated-word speech recognizer using a BNN. The techniques and schemes used in training and testing the network to improve the recognition rate, as well as to keep the system stable and the convergence rate high, worked well in the experiments reported in the thesis. However, this set of techniques may be only applied to this particular case. There are still no firm rules available to train the networks. Most of the rules of thumb are experiment dependent. More widely applicable learning and testing schemes are needed to further improve the recognition rates as well as increase the vocabulary size.

# APPENDIX A. SIMULATION PROGRAMS

```
% *****************************************************************
% Covarc.m     Covariance method to solve the leastsquare
% normal equation (X'X)a=(S,0)'
%
% Usage: [A,S]=covarc(x,p,N);
%        signal x of length N and order p
%
% *****************************************************************
function [A,S]=covarc(x,p,N);
for i = 1:(N-p)
   for j = 1:(p+1)
      xp(i,j) = x(p+1+i-j);
   end
end
Rx = xp'*xp;
[m,n] = size(Rx);
Rxp = [Rx(2:m,2:n)];
A = [inv(Rxp)*(-Rx(2:m,1))];
S = Rx(1:n)*[1;A];
S = S/N;


% *****************************************************************
% stef.m    Short time energy function
%
% Usage : y=stef(x);
%         x is input speech vector and y is corresponding
%         y corresponding short Time Energy function En.
%
% ************************************************************
function xos=stef(xi)
[m,n]=size(xi);
if n==1,
   xi=xi';              % make sure input is a row vector
end;
N= input (' enter window size (50 to 300 samples) N ? ');
% N=50 is used ;
```

```
wt=input (' choose window type 0. Rectangular 1. Hamming ? ');
% wt=1; hamming window is used
if wt==0
  window=ones(1,N);advance=fix(N/2);wtstrng=' Rect. Win';
else
  window=hamming (N)';advance=fix(N/4);wtstrng=' Hamm. Win';
end
%shortseq = 0;
if length(xi)<N          % case if length of input<length of window
  xi=[xi,zeros(1,N-length(xi))]; % zero pad
end
input=xi.^2;
imp_response=window.^2;
xo(1)=sum(input(1:N).*imp_response);
for n=advance:advance:length(xi)-N,
  xo(1+n/advance)=sum(input(n:n+N-1).*imp_response);
end;
xos=sum(xo);
subplot(211)
clg; plot(xo);
xlabel('time');title(['N=',num2str(N),wtstrng]);
ylabel('En');grid;



% ***********************************************************
% function xos=stazcr(xi)
% stzcr.m     Short time average zero crossing rate
%
% usage : y=stazcr(x)
%   x   : input speech vector
%   y   : Short-Time Avarage Zero-Crossing Rate Zn
%
% *******************************************************
function xos=stazcr(xi)
[m,n]=size(xi);
if n==1,
  xi=xi';              % make sure input is a row vector
end;
%N= input (' enter window size (50 to 300 samples) N ? ');
N=50;                  % window length N=50
%wt=input (' choose window type 0. Rectangular 1. Hamming ? ');
wt=1;                  % hamming window is used
```

```
if wt==0
  window=ones(1,N);advance=fix(N/2);wtstrng=' Rect. Win';
else
  window=hamming (N)';advance=fix(N/4);wtstrng=' Hamm. Win';
end
if length(xi)<N          % case if length of input<length of window
  xi=[xi,zeros(1,N-length(xi))]; % zero pad sequence
end
input=abs( sign( [xi(2:length(xi)),0] ) - sign(xi) );
xo(1)=sum(input(1:N).*window);
for n=advance:advance:length(xi)-N,
     xo(1+n/advance)=sum(input(n:n+N-1).*window);
end;
xos=sum(xo);
subplot(211)
clg; plot(xo);
xlabel('time');title(['N=',num2str(N),wtstrng]);
ylabel('En');grid;



% ********************************************************
% voiceunv.m      Voiced and unvoied discrimination
%
% usage: vuv=voiceunv(data)
% function will return  0   if unvoiced
%                       1   if voiced
%                      .5   if in between
%
% ********************************************************
function vuv=voiceunv(data)
NSeg=10;
NSample=fix(length(data)/NSeg);
for indx=1:NSeg,
  xx = data((NSample/2)*(indx-1)+1:(NSample/2)*(indx+1));
  en(indx) = stef(xx);
end;
vuv=sum(en(1:NSeg));
```

```
% ***************************************************************
% ar12lin.m     12th order AR model coefficients (normalized)
%
% Usage:  arcoeff=ar12lin(data)
%
% ***************************************************************
function arcoefff=ar12lin(data)
p=12;       % order of AR coefficients
Nd = 2;
x=decimate(data, Nd);
NSeg=10;
NSample=fix(length(x)/NSeg);
for indx1=1:NSeg,
   xx = x(NSample*(indx1-1)+1:NSample*indx1);
   xx = dtrend(xx);
   xstore(:,indx1) = xx(:) ./ max(xx);
   [a,s]=covarc(xx,p,length(xx));
   aa(:,indx1) = a(:);
   astore(:,indx1) = a(:) ./ max(a);
   ss(indx1) = s;
   en(indx1) = stef(xx);
   zc(indx1)=stazcr(xx);
   vuv(indx1)=voiceunv(xx);
end;
vuv =vuv/max(vuv);
ss=ss/max(abs(ss));
en=en/max(abs(en));
zc=zc/max(zc);
for indx3=1:NSeg
   if vuv(indx3)<.2
      vuv(indx3)=0;
   elseif vuv(indx3)<.8
      vuv(indx3)=.5;
   else
      vuv(indx3)=1;
   end
end
arcol=aa(:);
scalef=max(abs(arcol));
arcoef=aa/scalef;
arcoeff=[arcoef;ss;en;zc;vuv];
arcoefff=arcoeff(:);
```

```
% ****************************************************************
% arcoef.m        AR coefficients output file
%
% ****************************************************************
coef = ar12lin(dont(:));
coef10(1,:) = [coef' 1 0 0 0 0 0 0 0 0 0];
coef = ar12lin(ask(:));
coef10(2,:) = [coef' 0 1 0 0 0 0 0 0 0 0];
coef = ar12lin(me(:));
coef10(3,:) = [coef' 0 0 1 0 0 0 0 0 0 0];
coef = ar12lin(to(:));
coef10(4,:) = [coef' 0 0 0 1 0 0 0 0 0 0];
coef = ar12lin(carry(:));
coef3(1,:)  = [coef' 1 0 0];
coef5(1,:)  = [coef' 1 0 0 0 0];
coef10(5,:) = [coef' 0 0 0 0 1 0 0 0 0 0];
coef = ar12lin(an(:));
coef10(6,:) = [coef' 0 0 0 0 0 1 0 0 0 0];
coef = ar12lin(oily(:));
coef3(2,:)  = [coef' 0 1 0];
coef5(2,:)  = [coef' 0 1 0 0 0];
coef10(7,:) = [coef' 0 0 0 0 0 0 1 0 0 0];
coef = ar12lin(rag(:));
coef3(3,:)  = [coef' 0 0 1];
coef5(3,:)  = [coef' 0 0 1 0 0];
coef10(8,:) = [coef' 0 0 0 0 0 0 0 1 0 0];
coef = ar12lin(like(:));
coef5(4,:)  = [coef' 0 0 0 1 0];
coef10(9,:) = [coef' 0 0 0 0 0 0 0 0 1 0];
coef = ar12lin(that(:));
coef5(5,:)  = [coef' 0 0 0 0 1];
coef10(10,:)= [coef' 0 0 0 0 0 0 0 0 0 1];
```

```
% ****************************************************************
% artr.m     AR training data file
%
% ****************************************************************
load madd0\madd0wrd;
arcoef;
arall3=coef3;
```

```
save madd03tr.nna coef3 /ascii
arall5=coef5;
save madd05tr.nna coef5 /ascii
arall10=coef10;
save madd01tr.nna coef10 /ascii

load mbom0\mbom0wrd;
arcoef;
arall3=[arall3;coef3];
save mbom03tr.nna coef3 /ascii
arall5=[arall5;coef5];
save mbom05tr.nna coef5 /ascii
arall10=[arall10;coef10];
save mbom01tr.nna coef10 /ascii

save ar3.nna arall3 /ascii
save ar5.nna arall5 /ascii
save ar10.nna arall10 /ascii
```

## LIST OF REFERENCES

1. NeuralWare, Inc., *Neural Computing,* documentation for the Neural Professional II Plus Neural Network simulation Software, 1991.

2. Gorman, R.P. and Sejnowski, T.J. "Learned Classification of Sonar Targets Using a Massively Parallel Network," IEEE Trans. Acous., Speech, and Sig. Processing, Vol. 36, No. 7, July 1998.

3. Webster, Willard P., "Artificial Neural Networks and their Application to Weapons," Naval Engineering Journal, v. 103, pp. 46-59, May 1991.

4. Hetcht-Nielsen, R., *Neurocomputing,* Addison-Wesley, 1990.

5. Rabiner, L.R. and Schafer, R.W., *Digital Processing of Speech Signals,* Prentice-Hall, 1978.

6. Mariani, J., "Recent Advances in Speech Processing," IEEE Int. Conf. Acous., Speech, and Signal Processing, Vol. 1, pp 429-440, 1989.

7. O'Shaughnessy, D., *Speech Communication,* Addison-Wesley Publishing Company, 1987.

8. NTIS, *TIMIT, CD ROM on Line Documentation for the DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus,* American Helix, October 1990.

9. Tom, Daniel M. and Tenorio, Fernando M., "Short Utterance Recognition Using a Network with Minimum Training," *Neural Networks,* Vol. 4, No. 6, pp 711-722, 1991.

## INITIAL DISTRIBUTION LIST

| | No. of Copies |
|---|---|
| 1. Defense Technical Information Center<br>Cameron Station<br>Alexandria, Virginia, 22304-6145 | 2 |
| 2. Library, Code 52<br>Naval Postgraduate School<br>Monterey, California, 93943-5000 | 2 |
| 3. Chairman, Code EC<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey, California, 93943-5000 | 1 |
| 4. Dr. Murali Tummala, Code EC/Tu<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey, California, 93943-5000 | 1 |
| 5. Dr. Charles Therrien, Code EC/Ti<br>Department of Electrical and Computer Engineering<br>Naval Postgraduate School<br>Monterey, California, 93943-5000 | 1 |
| 6. Dr. R. Madan (Code 1114SE)<br>Office of Naval Research<br>800 North Quincy Street<br>Arlington, Virginia 22217-5000 | 1 |
| 7. Mr. Samuel J. Frazier (Code SY84)<br>Naval Air Warfare Center<br>Patuxent River, Maryland 20670 | 1 |
| 8. Lt Chau G. Le<br>SUPSHIP San Francisco<br>San Francisco, CA 94124-2996 | 2 |